International Journal of Automotive Technology, Vol. 6, No. 2, pp. 107–117 (2005)

1229-9138/2005/020-04

DEVELOPMENT AND IMPLEMENTATION OF DISTRIBUTED HARDWARE-IN-THE-LOOP SIMULATOR FOR AUTOMOTIVE ENGINE CONTROL SYSTEMS

M. YOON[1], W. LEE[2] and M. SUNWOO[1]*

[1]Department of Automotive Engineering, Hanyang University, Seoul 133-791, Korea
[2]Department of Control and Instrumentation Engineering, Changwon National University, Gyeongnam 641-773, Korea

(Received 6 January 2004; Revised 16 April 2004)

ABSTRACT–A distributed hardware-in-the-loop simulation (HILS) platform is developed for designing an automotive engine control system. The HILS equipment consists of a widely used PC and commercial-off-the-shelf (COTS) I/O boards instead of a powerful computing system and custom-made I/O boards. The distributed structure of the HILS system supplements the lack of computing power. These features make the HILS equipment more cost-effective and flexible. The HILS uses an automatic code generation extension, REAL-TIME WORKSHOP® (RTW®) of MATLAB® tool-chain and RT-LAB®, which enables distributed simulation as well as the detection and generation of digital event between simulation time steps. The mean value engine model, which is used in control design phase, is imported into this HILS. The engine model is supplemented with some I/O subsystems and I/O boards to interface actual input and output signals in real-time. The I/O subsystems are designed to imitate real sensor signals with high fidelity as well as to convert the raw data of the I/O boards to the appropriate forms for proper interfaces. A lot of attention is paid to the generation of a precise crank/cam signal which has the problem of quantization in a conventional fixed time step simulation. The detection of injection/command signal which occurs between simulation time steps are also successfully compensated. In order to prove the feasibility of the proposed environment, a simple PI controller for an air-to-fuel ratio (AFR) control is used. The proposed HILS environment and I/O systems are shown to be an efficient tool to develop various control functions and to validate the software and hardware of the engine control system.

KEY WORDS : HILS (Hardware-in-the-loop simulation), ECU (Electronic control unit)

1. INTRODUCTION

Over the past few years, the volume of software especially for engine electronic control unit (ECU) has shown an almost exponential growth in range and complexity, because it must fulfill the customers' demands for power enhancement as well as the government's regulations on emissions reduction and on-board diagnosis. Due to the increasing complexity and the inter-relationship between the design of the processes and the design of the control system, computer-aided methods for modeling, simulation and the design are increasingly being required. Therefore, open loop tests have limited functionality for software testing because the developer cannot simulate the overall system feedback loops, i.e. the proper test of control system requires that all control loops are in place and closed.

Traditionally, control systems are developed and tested using open loop testers and prototype vehicles. This is not only expensive but also time consuming and inflexible. In contrast, a model based development environment provides a less expensive alternative where controlled repeatable tests can be performed in much shorter time periods. Moreover, the ECU hardware is often developed in parallel with the control algorithms, and the control engineers have little time to test and verify the control algorithms. If a virtual vehicle test environment were available, a significant part of software development, verification and system validation could be completed in a laboratory environment. Hardware-in-the-loop simulation (HILS) systems provide such a virtual environment for system validation and verification where some of the system components are real hardware, and the others are simulated to mimic hardware components.

From energy production, to aerospace and aeronautics, to robotics, automotive, naval, and defense, real-time simulation and support for hardware-in-the-loop modeling are increasingly recognized as essential tools for

*Corresponding author. e-mail: msunwoo@hanyang.ac.kr

design in these and other industries. In the automotive society, HILS is indeed becoming a standard option for speeding up the ECU development time, and for ECU quality assurance. Many successful HILS applications have been reported recently (Hanselmann, 1996, Maclay, 1997, Isermann *et al.*, 1999, Babbitt and Moskwa, 1999, Kendall and Jones, 1999, Linjama *et al.*, 2000). In order to aid development efforts of a large system, Stasko *et al.* (1998) generalized HILS method and expanded it to a versatile HILS laboratory. Recently, Boot *et al.* (1999) discussed HILS along with test automation techniques in order to effectively meet the demands created by the increasing complexity of the software and the resulting tests. Raman *et al.* (1999) and Linjama *et al.* (2000) provided requirements for the overall HILS, such as flexibility, fidelity, expandability, and other criteria.

However, many of the reported HILS equipment require a complex interface between the real system and the HILS hardware, as well as a series of expensive and special computational platforms and I/O interface hardware. As an alternative for these complicated and expensive HILS equipment, PC-based HILS environments have been studied (Pollini and Innocenti, 2000, Baracos *et al.*, 2001). These help developers to set up the experimental system more easily and to test the control system more efficiently. In this paradigm, Lee *et al.* (2003) reported a PC-based HILS platform in which an engine is replaced by the model executed on a real-time computer system and is developed for designing an automotive engine control system. The platform consists of a widely used PC and commercial-off-the-shelf (COTS) I/O boards. Furthermore, this environment can be integrated seamlessly into the modern development process including rapid control prototyping (RCP).

An engine has event-based feature as well as time-based ones. The event-based feature focuses on the causal relationships among external events and the actions performed by a event, i.e., on "why" something happens, whereas the time-based feature focuses on the timing of actions, i.e., on "when" something happens. These features make it difficult to simulate engine behaviors because the major part of engine I/Os should be synchronized with the engine event, such as intake, compression, expansion, and exhaust. Because of this event driven characteristic of an engine, the accuracy of crank signal generation is not guaranteed, i.e., the modeled output of a crank angle sensor is quantized by fixed step time especially at high engine speed.

This paper proposes a novel modeling and simulation method for solving the above problem and an integrated software-hardware solution to HILS platform of an SI engines. The contents covers the development of the off-line plant model, proper choice and treatment of an I/O interface, the steps involved in configuring the model to run in real-time, and the interfacing of an off-line model with an actual control system.

2. HARDWARE/SOFTWARE ARCHITECTURE

To run research oriented ECUs for engine management without using a real engine or vehicle, it is necessary to simulate the electrical signals of the sensors and actuators plausibly. In order to compose HILS, a powerful computing system is necessary which can emulate behaviors of the controlled process accurately. This system should be interfaced with the actual controller in real-time through the signals of sensors and actuators. In order to enhance the control system's productivity, the HILS should be easy to operate. In other words, this HILS should be seamlessly integrated in the modern development process. In this section, the hardware and software components of the simulator test bench for the engine control system are described.

The controlled object and the auxiliary components in the control loop are replaced by real-time simulations of their behavior. The only real component in the test setup may be the ECU itself. The real part of an actuator is not adopted in the developed HILS because the engine controller is obviously divided from the actuator, i.e. the controller does not contain an ignition coil and injector. Furthermore, as proposed by Baracos *et al.* (2001), affordable PC hardware is used as the computing platform for the high-performance electromechanical simulation.

Sufficiently complex system dynamics may overwhelm even the fastest processor. The solution is to distribute the simulation and perform parallel processing. The RT-LAB® environment enables distributed simulation and parallel processing over a PC cluster (RT-LAB User's Guide, 2000). RT-LAB® from Opal-RT Inc. is an industrial grade software package for engineers who use mathematical block diagrams for simulation, control and related applications, and is selected for the distributed simulation and advanced digital signal generation/capture of the HILS platform. The RT-LAB® is shown to be an effective tool for real-time simulations in several studies (Ozard and Desira, 2000, Papini and Baracos, 2000, Rabbath *et al.*, 2000, Rabbath *et al.*, 2001, Chiasson and Tolbert, 2002). The RT-LAB® can detect input events that occur between computation time steps and compensate any calculations in order to remove errors that would be introduced if the event were only detected at the next time step after the event occurred. This feature is also for signal generation. We will call these 'asynchronous event detection' and 'asynchronous signal generation' respectively.

The physical configuration of the simulation system is divided into the host and the target sides. On the host side, the engineer designs his or her model and runs off-line simulations. On the target side, several micropro-

cessors are connected via a high-speed communication network and serve in the execution of the distributed simulations. The real-time simulation is executed in the real-time mode with QNX® or Neutrino® real-time operating system (RTOS). Control of the simulation is provided by RT-LAB® which sits on the host computer and communicates with the target nodes via TCP/IP. RT-LAB® works with the standard SIMULINK® block diagram languages and its code generators to enable parallel, real-time execution of simulations and input/output interfacing. RT-LAB® handles the set up of all the communications, including real-time data exchanges between target processors. Figure 1 shows how the HILS model is integrated with controller and real-time computer hardware. It is subdivided into the following parts:

(a) Command station
(b) Target nodes (real-time computing system)
(c) Signal interface; sensors and actuators
(d) Electronic control unit (ECU)

2.1. Command Station

The Command Station is a PC workstation that operates under Windows NT, and serves as the user interface. The model realization is performed with MATLAB®/SIMULINK® on the command station in order to use all the benefits of a graphical simulation environment. It is widely used in industry and academia and is becoming a standard tool for a capable, cost-effective off-line simulation solution. MATHWORKS Inc. tool-chain has the extensions of STATEFLOW®, and REAL-TIME WORKSHOP® (RTW®). STATEFLOW® enables state transition, finite state machine type behaviors, to be integrated with the continuous and discrete time representations traditionally provided by SIMULINK®. RTW® generates optimized, portable, and customizable code from SIMULINK® models. It frees engineers from the tedious and error-prone task of writing code. The separated code is automatically generated from the off-line simulation model of the engine and downloaded from the command station to the real-time computer hardware, namely, the target nodes.

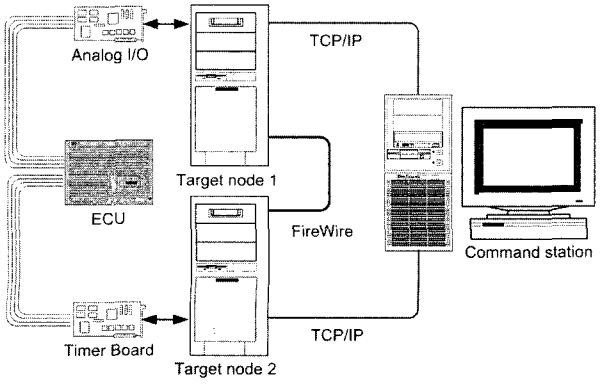

Figure 1. Hardware components of HILS.

A comfortable experimental environment is needed for the efficient use of the simulator. The simulator operation is performed with a user interface on the command station. All the relevant simulation quantities can be visualized on the command station or logged in real-time on target nodes. It also offers the possibility to regulate all the inputs or parameters of the model in the target nodes while the simulation is in progress. The visualization and parameter regulation are done via TCP/IP on Ethernet, which guarantees delivery and ordered queuing of transmitted packets. Therefore, both the reproducible and the interactive experiments can be performed through the command station.

2.2. Target Nodes

Powerful computer hardware is required to fulfill the real-time constraints for the hard real-time simulation of the HILS. Though there are plenty of powerful computational platforms as HILS hardware, it is convenient to use a commercial PC, as long as they can provide sufficient computing power to satisfy real-time constraints. The computer industry is growing exponentially, and high-performance affordable PCs are used for the low-cost HILS. Furthermore, a readily available environment saves the engineer a lot of time that would have otherwise been spent in becoming experienced.

In this study, the engine model is simulated on two Neutrino®-based, Pentium IV® 1.8 GHz processor with 512 MB RAM. The engine model is simulated on one node, while the crank/cam signal generation and spark/injection signal detection are dedicated to the other node. The target nodes are connected to the command station via the Ethernet adapter. These computers also include a real-time communication interface as well as I/O boards for accessing external equipment, ECU in this case. Because there are two QNX nodes, one of them is designated as the compilation node. The engine model and auxiliary subsystems are downloaded onto the one of target nodes (compilation node) and loaded after compilation, and then executed in real-time. Target nodes are connected together via the FireWire communication link; the I/O boards and target nodes are synchronized through this network.

2.3. Signal Interfaces

For HILS, some real components are located in the control loop; hence signal interfacing is needed. The identification and implementation of the interface between the computational platform and the controller is extremely important and very time consuming. The HILS environment requires signals from the computational platform to be interfaced with the hardware in the loop.

During the design phase, one must decide on the proper portioning of signal processing into available software and hardware. With respect to interfacing, the lowest physical interfacing layer provides a complete test environment for final products, e.g., a custom hardware circuitry. However, it hinders debugging of the control algorithm and software validation because the use of a realistic actuator operating signal with a simulated process may require considerable effort to design the signal interface.

The developed HILS assumes no real actuators and sensors, and all the signals are interfaced in logic-level for the simulation, i.e., simulated output on the computing platform. The control-oriented models using logic-level interfacing provide adequate fidelity for control software validation. The engine model does not require any real operating signal to an actuator such as a voltage of primary/secondary windings of an ignition coil, or the current imposed on the injector solenoid. Moreover, because the logic-level signals are used as the interfaces between a controller and a simulated process, the interfaces require neither an extra actuator interface device nor signal conditioning. The signal interface units of the developed HILS consist of COTS I/O cards, which can be treated easily by the RT-LAB® block library. PCI-6703® (National Instruments, Inc.) is used for analog I/O, and PCIDCC20-P® (ICS Advent, Inc.) is used for timing I/O and clock synchronization. For the real-time communication between target nodes, FireBoard 400-OHCI (Unibrain, inc.) is used.

2.4. Electronic Control Unit

The electronic control system under development is a real component in the HILS. In the early design phase, hardware of this electronic control unit is sometimes unavailable. However, in order to shorten development period, control and software engineers should design the controller based on assumed specifications. Therefore, many engineers use ready-made microcontroller evaluation boards with simplified customized I/O systems. In this study, logic-level I/O signal interface is assumed, and it helps control and software engineers to develop the controller with these evaluation boards. MPC555® of Motorola Inc. is selected as the control unit's microcontroller, and has a lot of characteristics which alleviate implementation difficulties. A Power-PC® core of MPC555® enables a model-based control due to floating-point calculation capability, and its sophisticated sub-modules help code implementation (MPC555 User's Manual, 1999).

3. HILS OF SI ENGINES

There are many stages in the typical development cycle of an engine control system. A number of steps in this development process require a mathematical model of the engine for which the controller is being developed. However, the scope and the intended usage of the model may differ from one step to the other. When the model is used in the core control algorithm development stage, a lot of hardware may not even exist at this stage and they are modeled very simply or omitted for the convenience of simulation. As the design matures, more detailed and complex models may be desired to test the controller. At the latter stages of the development process, the engine model is simulated in real-time with the controller module in a feedback loop. Typically, HILS adapts models that are used in the control system design phase, and these models are generally modified to fulfill several requirements such as inclusion of sufficient dynamics, exclusion of implicit equations or algebraic loops, and use of fixed step size with explicit simulation schemes, such as Euler, trapezoidal and Runge-Kutta integration methods.

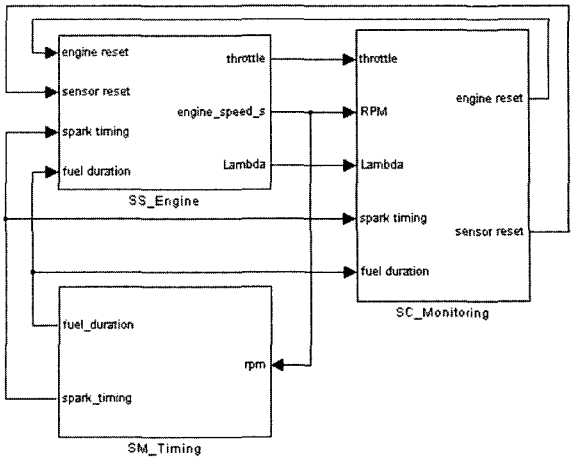

Figure 2. Overall model structure.

The model is decomposed for the use in RT-LAB® from within the SIMULINK® window. Since we use one host-processor and two target-processors, we chose to decompose the engine HILS model into 3 blocks (console, master and slave), as shown in Figure 2. The console block, *SC_Monitoring*, contains the display block and blocks for reset work which is useful for the correction of numerical error during simulation without halting the system. The master block, *SM_Timing*, contains the timing I/O subsystem including RT-EVENT blocks for the asynchronous event detect and the asynchronous signal generation. The remainder of the model has been placed in the slave block, *SS_Engine*. This block contains the engine model subsystem and the analog I/O subsystem.

The engine model and auxiliary subsystems used for the HILS are shown in Figure 3 and 4. The models have been developed in SIMULINK® and STATEFLOW® by

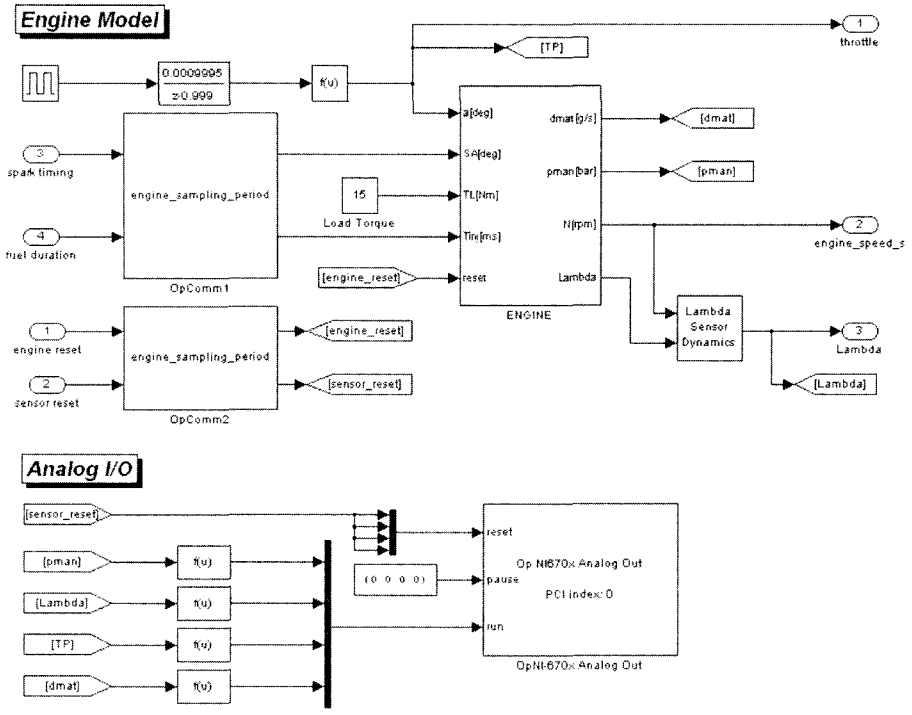

Figure 3. Engine HILS model (SS_Engine).

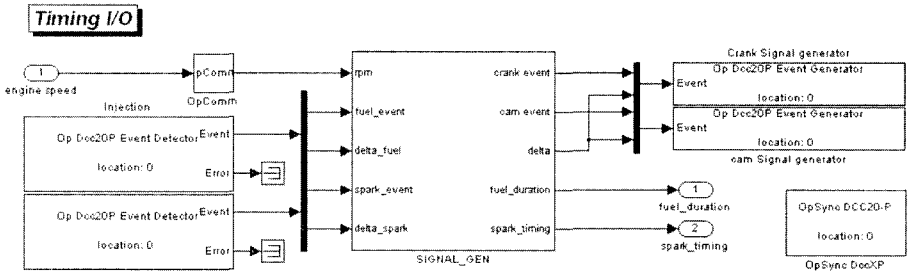

Figure 4. Engine HILS model (SM_Timing).

combining continuous time simulation blocks, finite-state machines, and textual components. The HILS model is decomposed hierarchically for the convenience of representation and management. The model is partitioned into an engine subsystem, an analog I/O subsystem, and a timing I/O subsystem. An engine model of the off-line simulation should be supplemented with other I/O signal modules to interface actual input and output signals in real-time. Analog signals, such as from a wide-band oxygen sensor, a coolant temperature sensor, pressure sensors, and others, are easily interfaced with the engine model through relatively simple algebraic equations. However, synchronization of the timing I/O signals, including a crankshaft signal, a camshaft signal, and spark and injection signals, with engine event makes the interface difficult. These problems are highlighted in the following sections, especially in subchapter 3.3.

3.1. Engine Subsystem

There have been many studies regarding the development of the control-oriented dynamic engine model (Yoon and Sumwoo, 1999, Dobner, 1983, Moskwa and Hedrick, 1992, Hendricks ans Sorenson, 1990, Powell and Cook, 1987). This study employs the engine model developed by Yoon and Sunwoo (1999). The engine model consists of three input variables (throttle angle, fuel flow rate, and spark timing), one disturbance (load torque), and three state variables (intake manifold pressure, engine speed, and fuel mass in the fuel film). The applicable operating range of the engine model is expanded by the inclusion of the change of minimum spark advance for best torque (MBT) with respect to the air-to-fuel ratio, and it also can be extended to lean-burn operations. This mathematical model is compact enough to run in real time, and its

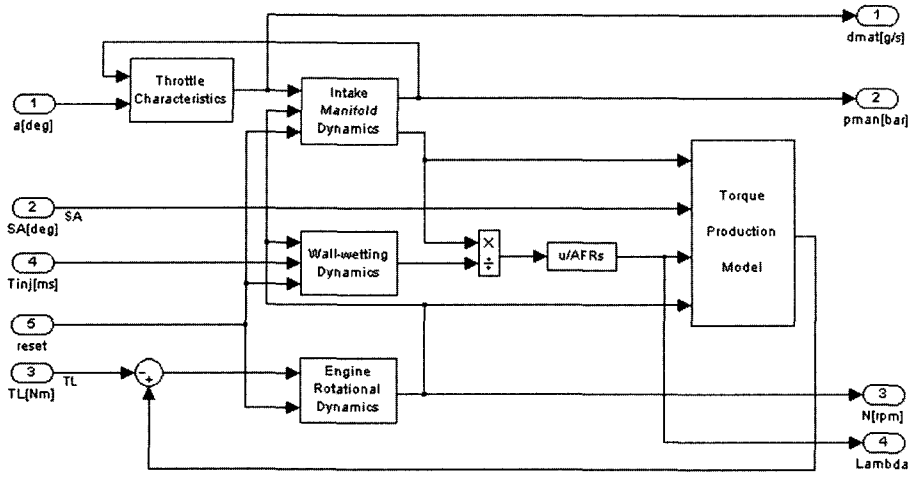

Figure 5. Engine model.

accuracy is guaranteed over a wide range of operating conditions.

The SIMULINK® implementation of the engine model is shown in Figure 5. The model consists of several subsystems, and this modularization makes user interface simple and improves the program module reusability and software productivity.

3.2. Analog I/O Subsystem

The role of the analog I/O subsystem (Figure 3) is to transform physical quantities of the model output into analog signals required by the engine controller. This includes manifold absolute pressure, oxygen sensor signal indicating air-fuel ratio (AFR), throttle position, and mass air flow rate. Several functions are used to imitate sensor characteristics and to generate the sensor signals with the D/A converter. The D/A converter is configured through some driver blocks of the RT-LAB® without hand-written codes.

3.3. Timing I/O Subsystem

This subsystem (Figure 4) provides a deliberate interface between time-based process and event-based control. Figure 6 shows the root level diagram of the timing I/O subsystem. The signal interfaces between HILS and ECU are accomplished through the counter/timer card. The timing I/O subsystem is comprised of a frequency modulation module, crank/cam signal generation module, and injection/spark signal measurement module.

The crankshaft angle signal and camshaft signal are

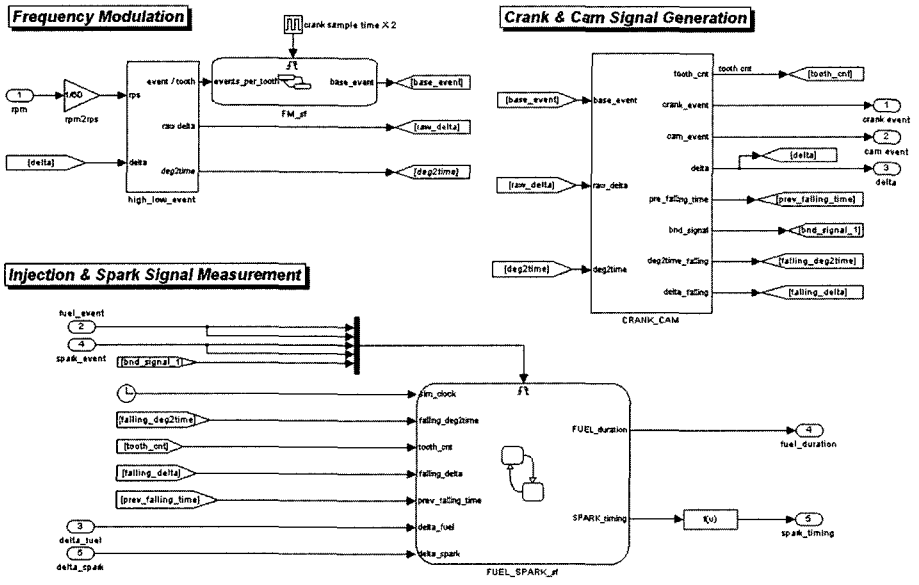

Figure 6. Timing I/O model.

used to synchronize an engine with an engine control system. The crankshaft angle and camshaft signals comprise a series of pulse train, which are normally generated by a toothed wheel and an magnetic induction sensors. The synchronized simulation of this kind of sensor causes demanding requirements on the HILS hardware. The HILS should generate the crankshaft and camshaft sensor signals according to changes in the engine speed, and detect control input signals with respect to the crankshaft position and simulation time base. In terms of a time based simulation method, the crankshaft signal can be thought of as a variable frequency signal source, and the camshaft signal as a reference position signal which marks the absolute position of angle. The spark and injection control signals should be measured based on these two signals: the crankshaft and camshaft signals.

When an event-based system is simulated in a time-based simulation environment, some timing problems may occur. In the case of an engine HILS, the timing of the crank/cam signals and the fuel/spark signals should be carefully considered in the engine model. Thus, the HILS should generate the crank and cam signals as real as possible, but a time-based simulator can change the state of these signals only at the simulation time step. Therefore, a timing discrepancy appears between the occurrence of a simulated event and that of a real one, and this aberration increases as the engine runs faster. Similar problems may also occur when the HILS measures the timing of the fuel and spark control signals. A time-based simulator cannot discern an event which occurs between the simulation time steps. This also causes a discrepancy between the timing of an actual event and the detection of the event by the HILS. These timing discrepancies degrade the accuracy of the HILS.

Generally, there exist two simulation techniques which are used to effectively simulate event-based systems in a time-base simulation environment. One approach to enhance the timing accuracy is to use the variable step size of the simulation time. However, because the variable step size solution cannot guarantee the termination time of each simulation step, it cannot be used to simulate a dynamic plant model in real-time. Another approach is to utilize a relatively short fixed simulation time step. This method generally needs more computing power than the variable step size approach does. In this case, a multi-rate simulation method is preferable to the single-rate one. By providing more computing power to the part which is executed in a small time step, the timing resolution of the simulation can be enhanced, and the computing power can be optimized (Lee and Yoon, 2003). Using the conventional time-based techniques, however, there are unavoidable differences between the simulated and the actual engine dynamics; this method cannot remove the problem, just relieve it.

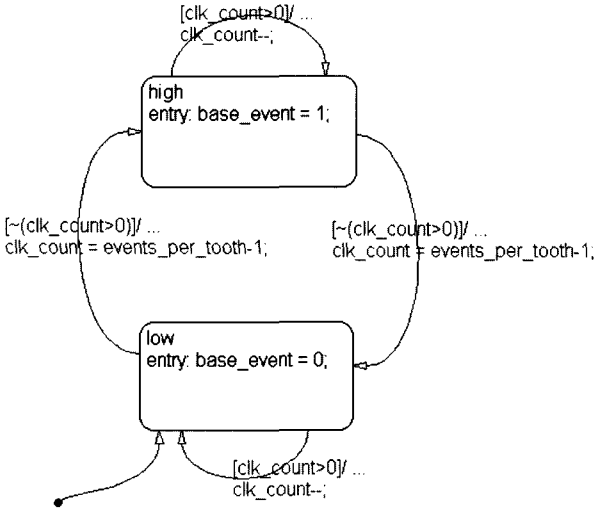

Figure 7. Frequency modulation module (STATEFLOW® diagram).

The solution for this problem, the crank signal is generated with new SIMULINK® blocks coming from the RT-EVENTS library (Rabbath *et al.*, 2000, RT-EVENT User's Guide, 2000). This library comprises a set of discrete-time blocks that use a compensated discrete-time simulation algorithm in order to account for the occurrence of discrete events in between fixed simulation steps. This avoids propagation of the error over time. The RT-EVENTS library contains blocks that compensate for errors introduced by asynchronous events, and is compatible with SIMULINK®, RTW® and RT-LAB®. In this study, "Event detector" and "Event generator" blocks are used (Figure 4).

3.3.1. Frequency modulation (FM) module

During the HILS experiment, all the timing I/O signals should be synchronized with the engine events. A base event (*base_evnet*) is generated by the *FM module,* and this pulse-train signal is used as a base signal for synchronization. The frequency of this signal is the function of the rotational speed of the crankshaft and the number of teeth. This frequency varies corresponding to the engine speed just like frequency modulation, and therefore this module is named the *FM module.* As shown in Figure 7, at each simulation time step, the engine model gives the engine speed information to the FM stateflow diagram (*FM_sf*), which generates pulses, *base_event*, at an appropriate frequency. The signal *raw_delta* in Figure 6, has the information in decimals, which are used for asynchronous event detection/signal generation and are ignored in conventional fixed step algorithms.

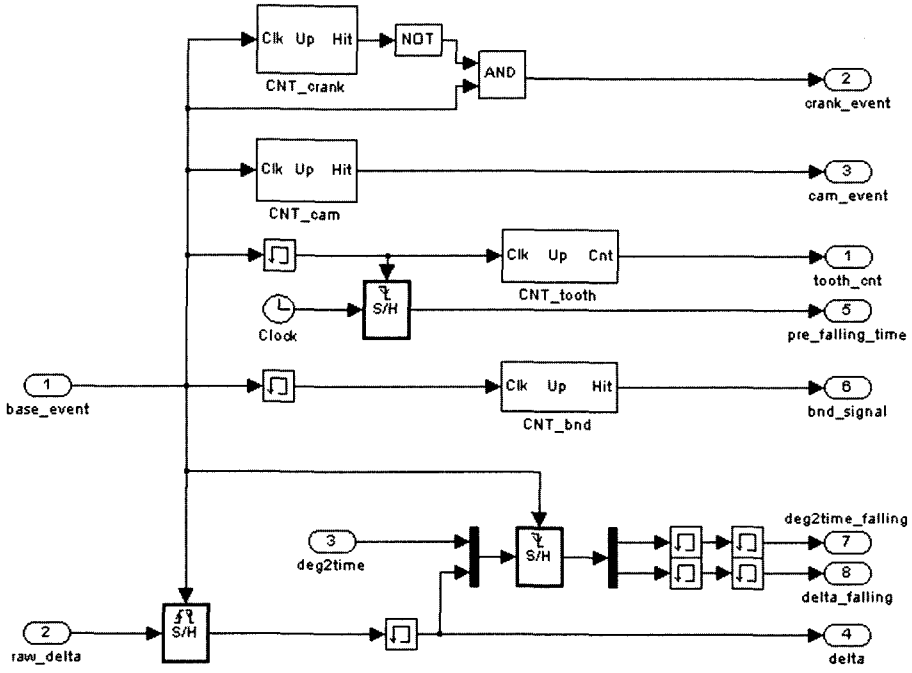

Figure 8. Crank & cam signal generation module.

3.3.2. Crank & cam signal generation module
CRANK_CAM contains four Counter blocks (*CNT_*) which count the signal of *base_event*. Several sample & hold and memory blocks are also used for data processing (Figure 8). For synchronization with the ECU, a crank signal should have one or two missing teeth, and a cam signal should be generated once per two revolutions of the crankshaft. *CNT_crank* generates a recognition signal where missing tooth should appears and the resulting signal, *crank_event*, is a more realistic crank event by adding the missing tooth effect to the base frequency-modulated signal (*base_event*). Note that the *crank_event* is not a crank signal itself, but an event signal for the operation of the counter/timer board. *CNT_cam* generates the signal for the cam event (*cam_event*) just as the *CNT_crank* generates the crank event. In order to indicate where the asynchronous signal generation should occur between fixed simulation steps, the value of *delta* is used and transmitted to the counter/timer board (Figure 4) after proper manipulation. Consequently, asynchronous signal generations for crank/cam signals are done by the Event generator block using *crank_event*, *cam_event* and *delta*.

Some ECU's can inject fuel many times over a certain range of the crankshaft angles. This range is called the *boundary angle* by Motorola (Mototola Low-Level Drivers User's Guide, 2000), and *CNT_bnd* generates *bnd_signal* for the Injection & spark signal measurement module, which indicates the allowable range of the crankshaft angle for additional fuel injection.

3.3.3. Injection & spark signal measurement module
The module for the injection-duration/spark-timing measurement, *FUEL_SPARK_sf*, is shown in Figure 9. Fuel injection duration (*FUEL_duration*) and spark advance (*SPARK_timing*) are calculated in *FUEL_SPARK_sf*. This module consists of two superstates: one is related to the fuel injection signals and the other is related to the spark control signals.

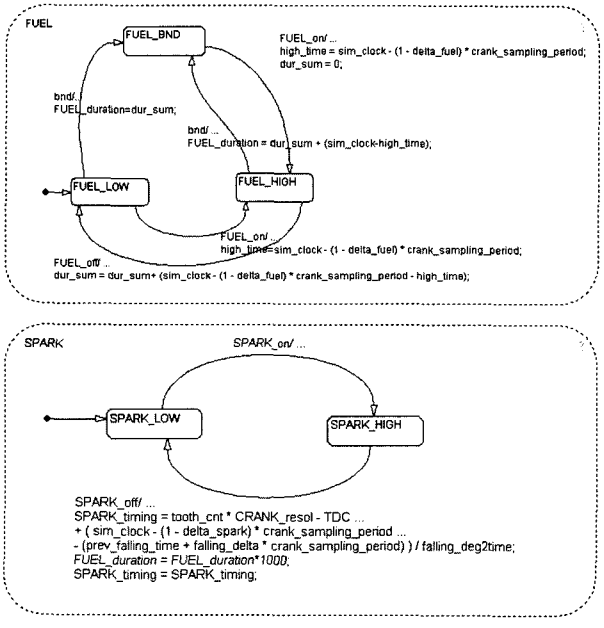

Figure 9. Fuel and spark signal measurement module.

The primary task of *FUEL* state is to measure the duration of the fuel injection pulses commanded by the ECU. The durations of all the fuel injection pulses are summed during the boundary angle, and the resultant summation (*FUEL_duration*) represents the total amount of the commended fuel. *SPARK* states check the transition of the spark control signal and measure the timing of the signal transition. A spark advance (*SPARK_timing*) is measured in the event-based manner. Furthermore, to more precisely reflect the event-based characteristics of the engine, the outputs of Event detector block (*delta_fuel* and *delta_spark*) are used for the compensation of fuel injection duration and spark advance, respectively (Figure 4).

3.4. Configuration

The simulation can be configured in an m-script file of MATLAB®. This configuration consists of an encoder related part and a simulation time related part. The number of encoder teeth and the number of missing teeth are needed to generate an encoder signal similar to the real signal, and the top dead center (TDC) position is also needed for synchronization with the controller.

RT-LAB® software runs on a hardware configuration consisting of the command station, compilation node, target nodes, the communication links (real-time and Ethernet), and the I/O boards. Simulations can be run entirely on the command Station computer, but they are typically run on one or more target nodes. For a real-time simulation, used real-time operating system (RTOS) for the target nodes is QNX. The Command Station and target nodes communicate with each other using communication links. For hardware-in-the-loop simulations, target nodes communicate with the ECU through I/O boards. Figure 2 shows that the simulation model is divided into three parts, i.e. the engine subsystem and the analog I/O subsystem is grouped into a relatively slow part (*SS_Engine*), while the timing I/O subsystem is grouped into a relatively fast part (*SS_Timing*). Each part has its own function and simulation time step: A fixed step of 400 μsec is used for the simulation of relatively slow parts, and 100 μsec for fast parts.

4. EXPERIMENTAL RESULTS

This section presents the experimental study on the HILS for an inline 4-cylinder DOHC engine in order to evaluate the applicability and performance of the simulator. The HILS equipment is composed of three general-purpose desktop PCs as shown Figure 10. One PC works as the command station, which provides a designer with the monitoring station, another is the target computer designated to asynchronous event detection and signal generation, and the third is the second target computer

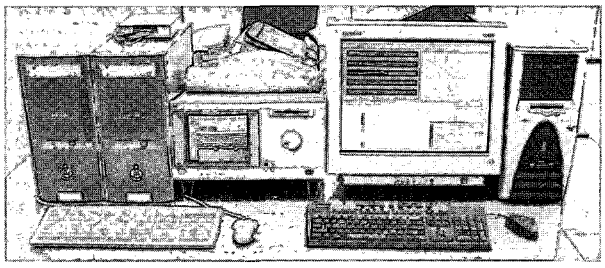

Figure 10. Photograph of the HILS equipment.

running engine model and analog I/O. Consequently, a set of target nodes operates like a real engine with the help of two COTS I/O boards.

A simple proportional-integral (PI) fuel injection control law for the air-to-fuel ratio (AFR) control is used in this experiment.

Figure 11 illustrates the output of the engine model and the controller, and the time trajectories of the signals are obtained via MAT files generated during the simulation run. The timing diagram of the I/O signals is shown in Figure 12. The throttle angle is changed to simulate a fast tip-in and tip-out situation. A low pass filter is used to consider time constant, which corresponds to a time scale on which an operator can change the throttle angle or a throttle-by-wire actuator can move the throttle plate. The wide-band lambda sensor is assumed to have band

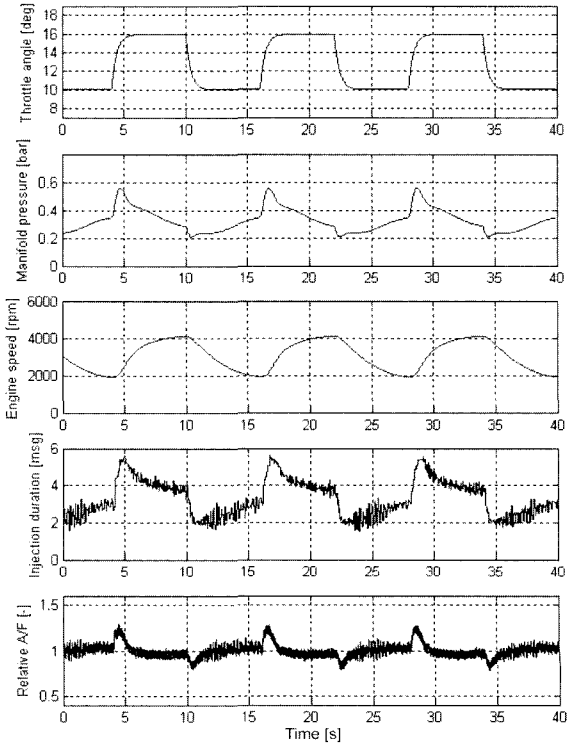

Figure 11. HILS experiment results.

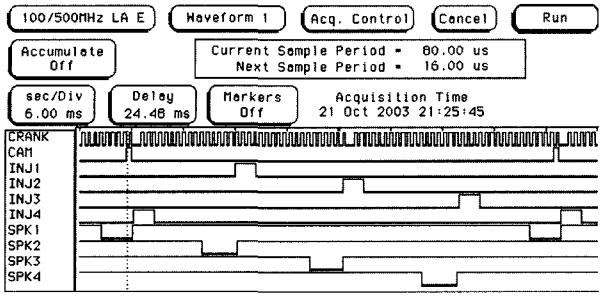

Figure 12. Timing diagram of closed-loop experiment.

limited white noise. An engine has a 36-toothed wheel with 1 missing tooth (*36-1 type*).

One can observe how the AFR controller decreases and increases the injection duration as a reaction to changes in the exhaust port AFR. The PI controller is executed every crank angle of 180 degree in an event-based manner. The controller calculates the amount of fuel injection, *INJ1~INJ4*, and the spark timing, *SPK1~SPK4*. The HILS system executes the engine model based on the input signals, and generates CRANK and CAM signal for the synchronization and other analog output signals, such as manifold pressure, AFR, throttle position, and mass air flow.

5. CONCLUSIONS

Many studies and examples show that HILS is an efficient tool for the development and testing of engine control systems. However, the use of this technique faces some challenging issues such as requirements for complex interfacing and an expensive computational platform, and execution of an accurate equipment simulation, which must operate in real-time.

A new PC-based HILS platform was developed for an automotive engine control system. This HILS equipment consists of a widely used PC and COTS I/O boards instead of powerful computing systems and custom-made I/O boards. These features make the HILS equipment more cost-effective and flexible. The HILS uses an automatic code generation extension, RTW® of MATLAB® tool-chain and this helps the control system developers to handle the controlled-object model more easily and to test the control system more comfortably and time-effectively. RT-LAB® enables model separation to allow distributed execution, while automatically generating, downloading, and running real-time distributed simulation.

A mean value engine model, which is used in control design phase, is imported in this HILS. The engine model is supplemented with some I/O subsystems and I/O boards to interface actual input and output engine signals in real-time. While the simulation is running at a regular time step, an event may occur between two time steps. If the simulation waits until the next time step to update the model, a numerical error will be introduced into the simulation. Using the RT-LAB® and proper timer board, one can accommodate this time difference and update the model appropriately. One can also generate digital events that occur between time steps. Using the special features of RT-LAB®, the I/O subsystems is designed to detect an event and generate a signal between simulation time step in order to synchronize exactly the status of the engine model with the control system, as well as to convert the raw data of the I/O boards to the appropriate forms for the interfacing.

To prove the feasibility of the proposed environment, a pilot project for the development of an AFR control system was performed. Based on the monitoring of relevant signals, it was shown that the solution provided relatively accurate simulations of the SI engine, i.e., the quantization of crank signal and inaccurate detection of injection/spark signals are removed. Furthermore, the engine model, analog I/O and timing I/O are successfully executed with the distribution. Consequently, the proposed HILS environment proved to be an efficient tool for developing new control functions and testing the software and hardware of engine control systems.

ACKNOWLEDGEMENT–This research is supported in part by MOST (Ministry of Science and Technology) under the National Research Laboratory (NRL) grant M1-0203-00-0058-02-J00-00-031-00, and part of the project 'Development of Partial Zero Emission Technology for Future Vehicle', and we are grateful for their financial suppot.

REFERENCES

Babbitt, R. and Moskwa, J. (1999). Implementation details and test results for a transient engine dynamometer and hardware in the loop vehicle model. *Proc. 1999 IEEE Int'l Sym. on CACSD,* Hawai'i, USA, 569–574.

Baracos, P., Murere, G., Rabbath, C. A. and Jin, W. (2001). Enabling PC-based HIL simulation for automotive applications. *Electric Machines and Drives Conference*, 721–729.

Boot, R., Richert, J. and Schutte, H. (1999). Automated test of ECUs in a hardware-in-the-loop simulation environment. *Proc. 1999 IEEE Int'l Sym. on CACSD*, Hawai'i, USA, 587–594.

Chiasson, J. N. and Tolbert, L. M. (2002). A library of simulink blocks for real-time control of HEV traction drives. *SAE Paper No.* 2002-01-1934.

Dobner, D. J. (1983). Dynamic engine models for control development – part I: nonlinear and linear model formulation. *Int. J. Vehicle Design*, Technological Advances in Vehicle Design Series, SP4.

Hanselmann, H. (1996). Hardware-in-the-loop simulation testing and its integration into a CACSD toolset. *Proc.*

1996 IEEE Int'l Sym. on CACSD, Dearborn, MI, 152–156.

Hendricks, E. and Sorenson, S. C. (1990). Mean value modeling of spark ignition engines. *SAE Paper No.* 900616.

Isermann, R., Schaffnit, J. and Sinsel, S. (1999). Hardware-in-the-loop simulation for the design and testing of engine-control systems. *Control Engineering Practice*, **7,** 643–653.

Kendall, I. R. and Jones, R. P. (1999). An investigation into the use of hardware-in-the-loop simulation testing for automotive electronic control systems. *Control Engineering Practice*, **7,** 1343–1356.

Lee, W., Yoon, M., and Sunwoo, M. (2003). A cost-and time-effective hardware-in-the-loop simulation platform for automotive engine control systems. *Proc. Institution of Mechanical Engineers*, **217,** *Part D: J. Automobile Engineering*, 41–52.

Linjama, M., Virvalo, T., Gustafsson, J., Lintula, J., Aaltonen, V. and Kivikoski, M. (2000). Hardware-in-the-loop environment for servo system controller design, tuning and testing. *Microprocessors and Microsystems*, **24,** 13–21.

Maclay, D. (1997). Simulation gets into the loop. *IEE Review*, **43,** 109–112.

Moskwa, J. J. and Hedrick, J. K. (1992). Modeling and validation of automotive engines for control algorithm development. *Trans. ASME, J. Dynamic Systems, Measurement and Control*, **114,** 278–285.

Motorola (2000). Motorola Low-Level Drivers User's Manual.

Motorola (1999). MPC555 User's Manual.

Opal-RT Technologies Inc. (2000). RT-EVENT User's Guide.

Opal-RT Technologies Inc. (2000). RT-LAB User's Guide.

Ozard, J. and Desira, H. (2000). Simulink model implementation on multi-processors under windows-NT. *Military, Government And Aerospace Simulation Symposium*, 197–204.

Papini, M. and Baracos, P. (2000). Real-time simulation, control and HIL with COTS computing clusters. *Modeling and Simulation Technologies*.

Pollini, L. and Innocenti, M. (2000). A synthetic environment for dynamic systems control and distributed simulation. *Control Systems Magazine, IEEE* **20, 2,** 49–61.

Powell, B. K. and Cook, A. (1987). Nonlinear low frequency phenomenological engine modeling and analysis. *Proc. American Control Conference*, 332–340.

Rabbath, C. A., Abdoune, M. and Belanger, J. (2000). Effective real-time simulations of event-based systems. *Simulation Conference, Proceedings. Winter*, **1,** 232–238.

Rabbath, C. A., Desira, H. and Butts, K. (2001). Effective modeling and simulation of internal combustion engine control systems. *Proc. American Control Conference*, **2,** 1321–1326.

Raman, S., Sivashankar, N., Milam, W., Stuart, W. and Nabi, S. (1999). Design and implementation of HIL simulators for powertrain control system software development. *Proc. American Control Conference*, San Diego, California, 709–713.

Stasko, J. C., Crandell, R. L., Dunn, M. T., Sureshbabu, N. and Weber, W. (1998). The versatile hardware-in-the-loop laboratory: beyond the ad-hoc fixture. *Proc. American Control Conference*, 508–512.

Yoon, P. J. and Sunwoo, M. (1999). A nonlinear dynamic engine modeling for controller design. *Trans. Korean Society Automotive Engineers* **7, 7,** 167–180.